Khronos Connects Software to Silicon

Open Consortium creating ROYALTY-FREE, OPEN STANDARD APIs for hardware acceleration

Defining the roadmap for low-level silicon interfaces needed on every platform

Graphics, compute, rich media, vision, sensor and camera processing

Rigorous specifications AND conformance tests for cross-vendor portability

Acceleration APIs BY the Industry FOR the Industry

Well over a BILLION people use Khronos APIs Every Day...
Significant Khronos API Ecosystem Advances

- **Vulkan API** - next generation graphics API
  - Low overhead, high-efficiency graphics and compute on GPUs
  - Formerly discussed as Next Generation OpenGL Initiative
  - Reveal and demos at GDC March 2015 - no formal specification yet

- **OpenCL 2.1 provisional specification released**
  - C++ Kernel language
  - Provisional specification released at GDC March 2015

- **SPIR-V** - first intermediate language for graphics and parallel computation
  - Will be used by both Vulkan AND OpenCL 2.1 core specifications
  - Provisional specification released at GDC March 2015
SPIR-V Transforms the Language Ecosystem

• Cross vendor intermediate representation
  - Language front-ends can easily access multiple hardware run-times
  - Acceleration hardware can leverage multiple language front-ends
  - Encourages tools for program analysis and optimization in SPIR form

• SPIR-V - first multi-API, intermediate language for parallel compute and graphics
  - Native representation for Vulkan shader and OpenCL kernel source languages

SPIR-V is a significant convergence point in the language ecosystem for graphics and parallel computation
Evolution of SPIR Family

- SPIR-V is first fully specified Khronos-defined SPIR standard
  - Does not use LLVM to isolate from LLVM roadmap changes
  - Includes full flow control, graphics and parallel constructs beyond LLVM
  - Khronos will open source SPIR-V <-> LLVM conversion tools

<table>
<thead>
<tr>
<th></th>
<th>SPIR 1.2</th>
<th>SPIR 2.0</th>
<th>SPIR-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLVM Interaction</td>
<td>Uses LLVM 3.2</td>
<td>Uses LLVM 3.4</td>
<td>100% Khronos defined Round-trip lossless conversion</td>
</tr>
<tr>
<td>Compute Constructs</td>
<td>Metadata/Iintrinsics</td>
<td>Metadata/Iintrinsics</td>
<td>Native</td>
</tr>
<tr>
<td>Graphics Constructs</td>
<td>No</td>
<td>No</td>
<td>Native</td>
</tr>
<tr>
<td>Supported Language Feature Set</td>
<td>OpenCL C 1.2</td>
<td>OpenCL C 1.2</td>
<td>OpenCL C 1.2 / 2.0</td>
</tr>
<tr>
<td></td>
<td>OpenCL C 2.0</td>
<td>OpenCL C 2.0</td>
<td>GLSL</td>
</tr>
<tr>
<td>OpenCL Ingestion</td>
<td>OpenCL 1.2 Extension</td>
<td>OpenCL 2.0 Extension</td>
<td>OpenCL 2.1 Core</td>
</tr>
<tr>
<td>Vulkan Ingestion</td>
<td>-</td>
<td>-</td>
<td>Vulkan Core</td>
</tr>
</tbody>
</table>
**SPIR-V at the Center of Language Ecosystem**

* Khronos considering developing open source implementations of these translators

**Other Intermediate Forms**

- **OpenCL C**
- **LLVM**
- **OpenCL C++**
- **GLSL**
- **New kernel and shader Languages**

**SPIR-V**
- 32-bit Word Stream
- Extensible and easily parsed
- Retains data object and control flow information for effective code generation and translation

**Other Languages**

- **OpenCL Driver A**
- **OpenCL Driver B**
- **Vulkan Driver X**
- **Vulkan Driver Y**

*Khronos considering developing open source implementations of these translators*
SPIR-V Advantages for Developers

- Developers can use same front-end compiler across multiple platforms
  - Eliminating major source of cross-vendor portability

- Reduces runtime shader/kernel compilation time
  - Driver only has to process SPIR-V not full source language

- Don’t have to ship shader/kernel source code
  - Provides a measure of IP protection

- Drivers are simpler and more reliable
  - No need to include front-end compilers

- SPIR-V Whitepaper
OpenCL - Portable Heterogeneous Computing

- Heterogeneous parallel programming of diverse compute resources
  - Targeting supercomputers -> embedded systems -> mobile devices
- One code tree can be executed on CPUs, GPUs, DSPs, FPGA and hardware
  - Dynamically interrogate system load and balance work across available processors
- OpenCL = Two APIs and Kernel language
  - C Platform Layer API to query, select and initialize compute devices
  - C Runtime API to build and execute kernels across multiple devices
OpenCL 2.1 Provisional Released!

• New OpenCL C++ kernel language based on a subset of C++14
  - Significantly enhanced programmer productivity and code performance

• Support for the new Khronos SPIR-V intermediate language in core
  - SPIR-V used to ingest from C++ front-end - no C++ compiler in driver
  - OpenCL C ingestion still supported to preserve kernel code investment

• Runs on any OpenCL 2.0-capable hardware
  - Only driver update required

OpenCL C++ Shading language
SPIR-V in Core
Subgroups into core
Subgroup query operations
clCloneKernel
Low-latency device timer queries

- Device partitioning
- Separate compilation and linking
- Enhanced image support
- Built-in kernels / custom devices
- Enhanced DX and OpenGL Interop

- Shared Virtual Memory
- On-device dispatch
- Generic Address Space
- Enhanced Image Support
- C11 Atomics
- Pipes
- Android ICD

<table>
<thead>
<tr>
<th>Spec</th>
<th>Release Date</th>
<th>Duration</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCL 1.0</td>
<td>Dec08</td>
<td>18 months</td>
<td>3-component vectors, Additional image formats, Multiple hosts and devices, Buffer region operations, Enhanced event-driven execution, Improved OpenGL data/event interop</td>
</tr>
<tr>
<td>OpenCL 1.1</td>
<td>Jun10</td>
<td>18 months</td>
<td>Device partitioning, Separate compilation and linking, Enhanced image support, Built-in kernels / custom devices, Enhanced DX and OpenGL Interop</td>
</tr>
<tr>
<td>OpenCL 1.2</td>
<td>Nov11</td>
<td>24 months</td>
<td></td>
</tr>
<tr>
<td>OpenCL 2.0</td>
<td>Nov13</td>
<td>16 months</td>
<td></td>
</tr>
<tr>
<td>OpenCL 2.1 (Provisional)</td>
<td>Mar15</td>
<td>18 months</td>
<td></td>
</tr>
</tbody>
</table>

© Copyright Khronos Group 2015 - Page 10
OpenCL as Parallel Language Backend

- WebCL
- Halide
- C++ AMP
- Aparapi
- Intel
- OpenACC
- Harlan

- JavaScript binding for initiation of OpenCL C kernels
- Language for image processing and computational photography
- MulticoreWare open source project on Bitbucket
- Embedded array language for Haskell
- Java language extensions for parallelism
- River Trail Language extensions to JavaScript
- Compiler directives for Fortran, C and C++
- PyOpenCL Python wrapper around OpenCL

Approaching 200 languages, frameworks and projects using OpenCL C as a compiler target to access vendor optimized, heterogeneous compute runtimes

This trend will accelerate with SPIR-V
OpenCL C++

• The OpenCL C++ kernel language is a static subset of C++14
  - Frees developers from low-level coding details without sacrificing performance

• C++14 features removed from OpenCL C++ for parallel programming
  - Exceptions, Allocate/Release memory, Virtual functions and abstract classes Function pointers, Recursion and goto

• Classes, lambda functions, templates, operator overloading etc..
  - Fast and elegant sharable code - reusable device libraries and containers
  - Templates enable meta-programming for highly adaptive software
  - Lambdas used to implement nested/dynamic parallelism

• C++11-based standard library optimized for data-parallel programming
  - Atomics, meta-programming & type traits, math functions...
  - Plus new library features: Work-item & Work-group functions, Dynamic parallelism, Image & Pipe functions...

Highly adaptive parallel software that delivers tuned performance across diverse platforms
OpenCL 2.1 API Enhancements

- Subgroup functionality moved into core with additional subgroup query operations
  - Expose hardware threads/warps/wavefronts and their cross-lane operations
  - Host queries for forward progress extension, and workgroup->subgroup mapping

- `clCloneKernel` enables copying of kernel objects and state
  - Safe implementation of copy constructors in wrapper classes
  - Used to pass kernel to second host thread, or for C++ wrappers for kernel objects

- Low-latency device timer queries
  - Support alignment of profiling data between device and host code

- `clCreateProgramWithIL` 
  - Enables ingestion of SPIR-V code by the runtime

- Priority and throttle hint extensions for queues
  - Specify execution priority on a per-queue basis

- Zero-size enqueue
  - Zero-sized dispatches are valid from the host
The Need for Vulkan

Ground-up design of a modern open standard API for driving high-efficiency graphics and compute on GPUs used across diverse devices

In the twenty two years since OpenGL was invented - the architecture of GPUs and platforms has changed radically

GPUs being used for graphics, compute and vision processing on a rapidly increasing diversity of platforms - increasing the need for cross-platform standards
Vulkan Explicit GPU Control

Complex drivers lead to driver overhead and cross vendor unpredictability

Error management is always active

Driver processes full shading language source

Separate APIs for desktop and mobile markets

Vulkan delivers the maximized performance and cross platform portability needed by sophisticated engines, middleware and apps
Cross Platform Challenge

• An explicit API that is also cross-platform needs careful design

One family of GPUs

One OS

One GPU on one OS

All Modern Platforms and GPUs

A challenge that needs...
Participation of key players
Proven IP Framework
Battle-tested cooperative model
The drive to not let the 3D industry fragment
Vulkan Layered Ecosystem

Applications can use Vulkan directly for maximum flexibility and control

Application uses utility libraries to speed development

Utility libraries and layers

Application

Games Engines fully optimized over Vulkan

Rich Area for Innovation
- Many utilities and layers will be in open source
- Layers to ease transition from OpenGL
- Domain specific flexibility

Developers can choose at which level to use the Vulkan Ecosystem
Vulkan Multi-threading Efficiency

1. Multiple threads can construct Command Buffers in parallel. Application is responsible for thread management and synch.

2. Command Buffers placed in Command Queue by separate submission thread.
Vulkan - Enhancing Driver Reliability

Streamlined API is easier to implement and test

Cross-vendor Portability

SPIR-V intermediate language improves shader program portability and reduces driver complexity

Open source conformance test source components for community engagement
Khronos is considering open sourcing compiler front-ends, such as GLSL to SPIR-V Translator. GLSL will be the first shading language supported by Vulkan. Game Engines can flexibly target SPIR-V and Vulkan back-ends. Future diversity in domain-specific languages, frameworks, and tools, such as E.g. C++ Shading Language, is also being considered. SPIR-V ingest is supported in the Vulkan core. The Vulkan Language Ecosystem supports Vulkan Runtime and devices such as Device X, Device Y, and Device Z.
Vulkan Tools Architecture

- Layered design for cross-vendor tools innovation and flexibility
  - IHVs plug into a common, extensible architecture for code validation, debugging and profiling during development without impacting production performance

- Common Loader used to enable use of tools layers during debug
  - Cross-vendor API calls provide debug data

Production Path (Performance)

- Vulkan-based Title
- Vulkan’s Common Loader
- IHV’s Installable Client Driver
- Debug Layers can be installed during Development
- Validation Layers

Interactive Debugger

Debug information via standardized API calls
Vulkan Tools Ecosystem

- Extensible modular architecture encourages many fine-grained layers - new layers can be added easily
- Khronos encouraging open community of tools e.g. shader debugging
- Valve, LunarG, Codeplay and others are already driving the development of open source Vulkan tools
- Customized interactive debugging and validation layers will be available together with first drivers
## Ground-up Explicit API Redesign

<table>
<thead>
<tr>
<th>OpenGL</th>
<th>Vulkan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Originally architected for graphics workstations with direct renderers and split memory</strong></td>
<td><strong>Matches architecture of modern platforms including mobile platforms with unified memory, tiled rendering</strong></td>
</tr>
<tr>
<td><strong>Driver does lots of work: state validation, dependency tracking, error checking. Limits and randomizes performance</strong></td>
<td><strong>Explicit API – the application has direct, predictable control over the operation of the GPU</strong></td>
</tr>
<tr>
<td><strong>Threading model doesn’t enable generation of graphics commands in parallel to command execution</strong></td>
<td><strong>Multi-core friendly with multiple command buffers that can be created in parallel</strong></td>
</tr>
<tr>
<td><strong>Syntax evolved over twenty years – complex API choices can obscure optimal performance path</strong></td>
<td><strong>Removing legacy requirements simplifies API design, reduces specification size and enables clear usage guidance</strong></td>
</tr>
<tr>
<td><strong>Shader language compiler built into driver. Only GLSL supported. Have to ship shader source</strong></td>
<td><strong>SPIR-V as compiler target simplifies driver and enables front-end language flexibility and reliability</strong></td>
</tr>
<tr>
<td><strong>Despite conformance testing developers must often handle implementation variability between vendors</strong></td>
<td><strong>Simpler API, common language front-ends, more rigorous testing increase cross vendor functional/performance portability</strong></td>
</tr>
</tbody>
</table>
Vulkan Drawing - Example

- Command Buffers greatly reduce CPU time when being re-used
- Replace thousands of traditional OpenGL calls with just one
- Keeps flexibility to still manipulate data afterwards (vertices, transforms, materials...)

```c
glBindBufferBase (UBO, 0, uboView);
foreach (obj in scene) {
    glBindVertexBuffer (0, obj.geometry->vbo, 0, vtxSize);
    glBindBuffer (ELEMENT, obj.geometry->ibo);
    glBindBufferRange (UBO, 1, uboMatrices, obj.mtxOffset, mtxSize);
    glUseProgram (progSolid);
    foreach (batch in obj.batches) {
        glBindBufferRange (UBO, 2, uboMaterial, batch.mtlOffset, mtlSize);
        glMultiDrawElements (TRIANGLES ...);
    }
    glUseProgram (progEdges);
    ...
}
```

Thousand of state changes and 10s of thousands drawcalls

```c
vkQueueSubmit (... , 1 , &cmdbuffer, ...);```

© Copyright Khronos Group 2015 - Page 24
Vulkan Status

• Rapid progress since project start in June 2014
  - Significant proposals and IP contributions received from members

• Participants come from all segments of the graphics industry
  - Including an unprecedented level of participation from game engine ISVs

• Initial specs and implementations expected this year
  - Will work on any GPU hardware that supports OpenGL ES 3.1 and up
  - Can ship on any OS - including previous versions of Windows
Khronos Open Standards for Graphics and Compute
A comprehensive family of APIs to address the full spectrum of developer requirements

1990’s | Workhorse cross-platform API for professional 3D apps and gaming

2000’s | Ubiquitous API for mobile gaming and general purpose graphics

2008 | Heterogeneous parallel computation

2014 | Portable intermediate representation for graphics and parallel compute

2015 | High-efficiency GPU graphics and compute API for performance critical apps

All APIs will be evolved and maintained to meet industry needs
Rich mix of open technologies for future innovation
Virtual Reality Will Influence Graphics APIs

• The ability to generate ‘Presence’ is becoming achievable at reasonable cost
  - Using visual input to generate subconscious belief in a virtual situation

• PC-based AND mobile systems
  - Beginning to enable developer experimentation

• VR Requirements will affect how graphics APIs generate visual imagery
  - Control over generation of stereo pairs - slightly different view for each eye
  - Optical system geometric correction in rendering path
  - Reduce latency through elimination of in-driver buffering
  - Asynchronously warp framebuffer for instantaneous response to head movement

Samsung Gear VR
Vision Pipeline Challenges and Opportunities

Growing Camera Diversity
Flexible sensor and camera control to GENERATE an image stream

Diverse Vision Processors
Use efficient acceleration to PROCESS the image stream

Sensor Proliferation
Combine vision output with other sensor data on device

Khronos coordinating with MIPI on camera control and data formats
Vision Processing Power Efficiency

• Wearables will need ‘always-on’ vision
  - With smaller thermal limit / battery than phones!

• GPUs have x10 imaging power efficiency over CPU
  - GPUs architected for efficient pixel handling

• Dedicated Hardware/DSPs can be even more efficient
  - With some loss of generality

• Mobile SOCs have space for more transistors
  - But can’t turn on at same time = Dark Silicon
  - Can integrate more gates ‘for free’ if careful how and when they are used

Potential for dedicated sensor/vision silicon to be integrated into Mobile Processors
But how will they be programmed for PORTABILITY and POWER EFFICIENCY?
OpenVX - Power Efficient Vision Acceleration

- Out-of-the-Box vision acceleration framework
  - Enables low-power, real-time applications
  - Targeted at mobile and embedded platforms
- Portability across diverse vision HW
  - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs ...
- Enables low-power implementations
  - E.g. low-power CPU can manage dedicated low-power vision hardware and DSPs
OpenVX Graphs - The Key to Efficiency

- OpenVX enables developer to express a graph of image operations (‘Nodes’)
  - Enables execution optimizations for power and performance efficiency

- Implementer can use diverse optimization methods
  - Each Node can be implemented in software or accelerated hardware
  - Nodes may be fused by the implementation to eliminate memory transfers
  - Processing can be tiled to keep data entirely in local memory/cache

Example OpenVX Graph
OpenVX 1.0 Functional Overview

• Core data structures
  - Images and Image Pyramids
  - Processing Graphs, Kernels, Parameters

• Image Processing
  - Arithmetic, Logical, and statistical operations
  - Multichannel Color and BitDepth Extraction and Conversion
  - 2D Filtering and Morphological operations
  - Image Resizing and Warping

• Core Computer Vision
  - Pyramid computation
  - Integral Image computation

• Feature Extraction and Tracking
  - Histogram Computation and Equalization
  - Canny Edge Detection
  - Harris and FAST Corner detection
  - Sparse Optical Flow

OpenVX 1.0 defines framework for creating, managing and executing graphs
Focused set of widely used functions (Nodes) to be accelerated
Wideley used extensions adopted into future versions of the core
Implementers can add functions as extensions

Is Extensible
Khronos maintains extension registry
Example Graph - Stereo Machine Vision

Tiling extension enables user nodes (extensions) to also optimally run in local memory
OpenVX and OpenCV are Complementary

<table>
<thead>
<tr>
<th>Governance</th>
<th>Community driven open source with no formal specification</th>
<th>Formal specification defined and implemented by hardware vendors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conformance</td>
<td>No conformance tests for consistency and every vendor implements different subset</td>
<td>Full conformance test suite / process creates a reliable acceleration platform</td>
</tr>
<tr>
<td>Portability</td>
<td>APIs can vary depending on processor</td>
<td>Hardware abstracted for portability</td>
</tr>
<tr>
<td>Scope</td>
<td>Very wide</td>
<td>Tight focus on hardware accelerated functions for mobile vision</td>
</tr>
<tr>
<td></td>
<td>1000s of imaging and vision functions</td>
<td>Use external camera API</td>
</tr>
<tr>
<td></td>
<td>Multiple camera APIs/interfaces</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>Memory-based architecture</td>
<td>Graph-based execution</td>
</tr>
<tr>
<td></td>
<td>Each operation reads and writes memory</td>
<td>Optimizable computation, data transfer</td>
</tr>
<tr>
<td>Use Case</td>
<td>Rapid experimentation</td>
<td>Production development &amp; deployment</td>
</tr>
</tbody>
</table>

© Copyright Khronos Group 2015 - Page 34
OpenVX Status

- Finalized OpenVX 1.0 specification released October 2014
  - [www.khronos.org/openvx](http://www.khronos.org/openvx)

- Full conformance test suite and Adopters Program available
  - $20K Adopters fee ($15K for members) - working group reviews submitted results
  - Test suite exercises graph framework and functionality of each OpenVX 1.0 node
  - Approved Conformant implementations can use the OpenVX trademark

- Khronos working on sample implementation of OpenVX 1.0
  - Will be released as open source
Khronos APIs for Vision Processing

• Any compute API can be used for vision acceleration
  - OpenCL, OpenGL Compute Shaders …

• OpenVX is a Khronos vision API that does not NEED a CPU/GPU complex
  - Can use any processor – from high-end GPU, through DSPs to hardware blocks

• The higher abstraction level of OpenVX protects app from hardware differences
  - Enables low-power, always-on acceleration – with application portability

Many implementers may choose to use OpenCL or OpenGL Compute Shaders to implement OpenVX nodes and OpenVX to enable a developer to connect those nodes into a graph.
Need for Camera Control API - OpenKCAM

- Advanced control of ISP and camera subsystem - with cross-platform portability
  - Generate sophisticated image stream for advanced imaging & vision apps
- No platform API currently fulfills all developer requirements
  - Portable access to growing sensor diversity: e.g. depth sensors and sensor arrays
  - Cross sensor synch: e.g. synch of camera and MEMS sensors
  - Advanced, high-frequency per-frame burst control of camera/sensor: e.g. ROI
  - Multiple input, output re-circulating streams with RAW, Bayer or YUV Processing

OpenKCAM standard is still in development

Image Signal Processor (ISP)

Defines control of Sensor, Color Filter Array
Lens, Flash, Focus, Aperture

Auto Exposure (AE)
Auto White Balance (AWB)
Auto Focus (AF)

Stream of Images for Vision Processing

OpenVX™
OpenKCAM is FCAM-based

- FCAM (2010) Stanford/Nokia, open source
- Capture stream of camera images with precision control
  - A pipeline that converts requests into image stream
  - All parameters packed into the requests - no visible state
  - Programmer has full control over sensor settings for each frame in stream
- Control over focus and flash
  - No hidden daemon running
- Control ISP
  - Can access supplemental statistics from ISP if available
- No global state
  - State travels with image requests
  - Every pipeline stage may have different state
  - Enables fast, deterministic state changes

Khronos coordinating with MIPI on camera control and data formats
Sensor Industry Fragmentation ...
Low-level Sensor Abstraction API

Apps Need Sophisticated Access to Sensor Data
Without coding to specific sensor hardware

Apps request semantic sensor information
StreamInput defines possible requests, e.g.
Read Physical or Virtual Sensors e.g. “Game Quaternion”
Context detection e.g. “Am I in an elevator?”

Sensor Discoverability
Sensor Code Portability

StreamInput processing graph provides optimized sensor data stream
High-value, smart sensor fusion middleware can connect to apps in a portable way
Apps can gain ‘magical’ situational awareness

Advanced Sensors Everywhere
Multi-axis motion/position, quaternions, context-awareness, gestures, activity monitoring, health and environmental sensors

StreamInput™
Khronos APIs for Augmented Reality

AR needs not just advanced sensor processing, vision acceleration, computation and rendering - but also for all these subsystems to work efficiently together.

- Advanced Camera Control and stream generation
- MEMS Sensors
- Precision timestamps on all sensor samples
- EGLStream - stream data between APIs
- Vision Processing
- Sensor Fusion
- 3D Rendering and Video Composition On GPU
- Application on CPUs, GPUs and DSPs
- Audio Rendering

© Copyright Khronos Group 2015 - Page 41
WebGL Ecosystem

Content downloaded from the Web
Content
JavaScript, HTML, CSS, ...

JavaScript Middleware

Middleware provides accessibility for non-expert programmers
E.g. three.js library

Browser provides WebGL 3D engine alongside other HTML5 technologies - no plug-in required

OS Provided Drivers
WebGL uses OpenGL ES 2.0 or Angle for OpenGL ES 2.0 over DX9

Low-level APIs provide a powerful foundation for a rich JavaScript middleware ecosystem

Reliable WebGL relies on work by both GPU and Browser Vendors
-> Khronos has the right membership to enable that cooperation
Pervasive WebGL

- WebGL on EVERY major desktop and mobile browser
- Portable (NO source change) 3D applications are possible for the first time

---

**WebGL - 3D Canvas graphics**

Method of generating dynamic 3D graphics using JavaScript, accelerated through hardware.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Current</th>
<th>Usage</th>
<th>Global</th>
<th>51.45% + 26.86% = 78.31%</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android Browser*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome for Android</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://caniuse.com/#feat=webgl
WebGL Tool/Engine Ecosystem

Epic Citadel - WebGL HTML 5 Benchmark (Firefox 22)

https://www.youtube.com/watch?v=l9KRBuVBjVo
WebGL Content Creation is Still Hard

- Tools, engines and open source libraries use proprietary formats
  - Over thirty 3D formats in use - and counting

- Common formats such as OBJ/STL can only contain single-models NOT scenes
  - Need lights, cameras, animations, scene hierarchy etc.

- COLLADA is feature-rich - BUT not designed as transmission format
  - Meshes, materials, textures, scene hierarchy, cameras, lights, animations...
  - Files are text-based XML - big to download and slow to parse

- Result - WebGL content teams roll tools and formats for each new project
3D Needs a Transmission Format!

- Need to bridge the gap between tools and today’s GL based apps
  - Reduce duplicated effort in content pipelines
  - Common publishing format for content tools and services

- Browsers support loading of standard formats for many media types
  - With compression - for fast network transmission
  - With efficient client decode - especially important on mobile devices

<table>
<thead>
<tr>
<th>Audio</th>
<th>Video</th>
<th>Images</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP3</td>
<td>H.264</td>
<td>JPEG</td>
<td>?</td>
</tr>
<tr>
<td>napster</td>
<td>YouTube</td>
<td>facebook</td>
<td>!</td>
</tr>
</tbody>
</table>

An effective and widely adopted codec ignites previously unimagined opportunities for a media type
glTF = “JPEG for 3D”

- ‘GL Transmission Format’
  - Runtime asset format for WebGL, OpenGL ES, and OpenGL applications

- Compact representation for download efficiency
  - Binary mesh and animation data
  - Extension capability for future formats with compression and streaming

- Loads quickly into memory
  - JSON for scene structure and other high-level constructs
  - GL native data types require no additional parsing

- Full-featured
  - 3D constructs (hierarchy, cameras, lights, common materials, animation)
  - Full support for shaders and arbitrary materials

- Runtime Neutral
  - Can be created and used by any tool, app or runtime
COLLADA and glTF Ecosystem

OpenCOLLADA Importer/Exporter.
Open sourced COLLADA Conformance Tests

PLAYCANVAS
Web-based Tools

Khronos is developing standards-based 3D ‘tools to playback’ pipeline in open source

COLLADA2GLTF Translator
Autodesk creating FBX to glTF Translator!!

WebGL deployment

Three.js glTF Importer.
Rest3D initiative
glTF Adoption

three.js loader
https://github.com/mrdoob/three.js/

rest3d viewer
https://github.com/amd/rest3d

Cesium Engine
https://github.com/AnalyticalGraphicsInc/cesium

Montage Viewer
https://github.com/fabrobinet/glTF-webgl-viewer

LibreOffice
https://www.libreoffice.org/
glTF Status and Resources

- Open specification - Open process
  - Spec, and sample code: [https://github.com/KhronosGroup/glTF](https://github.com/KhronosGroup/glTF)
  - All features backed up by multiple implementations in code
  - Aiming for ratified glTF 1.0 at SIGGRAPH 2015

- COLLADA2GLTF open-source converter is gaining robustness and momentum
  - [https://github.com/KhronosGroup/glTF/tree/master/converter/COLLADA2GLTF](https://github.com/KhronosGroup/glTF/tree/master/converter/COLLADA2GLTF)
  - Binaries are available on GitHub for easy use

- Three.js glTF loader
  - [https://github.com/KhronosGroup/glTF/tree/master/loaders/threejs](https://github.com/KhronosGroup/glTF/tree/master/loaders/threejs)
  - Most glTF features are already supported

- OpenCOLLADA - import/export plugins for Max and Maya
  - On Windows, Linux and Mac OSX

- FBX-glTF converter being developed by Autodesk!!!
  - [https://github.com/cyrillef/FBX-glTF](https://github.com/cyrillef/FBX-glTF)

- glTF working group working on direct Unity exporter...
glTF and Compression Extension

- Benchmarking 3D compression formats for potential glTF extensions
  - Baseline is GZIP
- MPEG royalty-free Scalable Complexity compresses Meshes, Skinning, Animations
  - 3D Mesh Compression codec MPEG-SC3DMC
  - Open3DGC JavaScript and C/C++ implementation
  - WebGL Open3DGC loader https://github.com/fabrobinet/glTF-webgl-viewer
- WebGL-loader is Google lightweight compression format for WebGL content

<table>
<thead>
<tr>
<th>Format</th>
<th>CAD Models (Mbytes)</th>
<th>3D Scanned Models (Mbytes)</th>
<th>MPEG dataset (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ</td>
<td>1310 (100%)</td>
<td>736 (100%)</td>
<td>600 (100%)</td>
</tr>
<tr>
<td>Gzip</td>
<td>336 (26%)</td>
<td>204 (28%)</td>
<td>157 (26%)</td>
</tr>
<tr>
<td>Webgl-loader</td>
<td>219 (17%)</td>
<td>117 (16%)</td>
<td>103 (17%)</td>
</tr>
<tr>
<td>Webgl-loader + Gzip</td>
<td>80 (6%)</td>
<td>38 (5%)</td>
<td>26 (4%)</td>
</tr>
<tr>
<td>Open3DGC</td>
<td>67 (5%)</td>
<td>22 (3%)</td>
<td>22 (4%)</td>
</tr>
</tbody>
</table>

Open3DGC is 5x-9x more efficient than GZIP and 1.2x-1.5x more efficient than webgl-loader
Call to Action

• More detailed information on new specifications
  - https://www.khronos.org/spir
  - https://www.khronos.org/opencl/
  - https://www.khronos.org/vulkan

• Khronos seeking feedback on Vulkan, SPIR and OpenCL 2.1 Forums
  - https://www.khronos.org/spir_v_feedback_forum
  - https://www.khronos.org/opencl/opencl_feedback_forum
  - https://www.khronos.org/vulkan/vulkan_feedback_forum

• Any company or organization is welcome to join Khronos for a voice and a vote in any of these standards
  - www.khronos.org
  - ntrevett@nvidia.com
  - @neilt3d